

A RUST BASED, ECS PATTERNED, INTERACTIVE FLIGHT SIMULATOR

THESIS

Chad A. Willis, 2d Lt, USAF

AFIT-ENG-MS-21-M-XXX

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

###### AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A.

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

(IF your document is limited, place your Destruction Notice Here)

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.AFIT-ENG-MS-21-M-XXX

A RUST BASED, ECS PATTERNED, INTERATIVE FLIGHT SIMULATOR

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in xxxxx

Chad A. Willis, BS

2d Lt, USAF

March 2021

DISTRIBUTION STATEMENT A.  
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENG-MS-21-M-XXX

A Rust based, ecs patterned, interactive flight simulator

THESIS

Chad A. Willis, BS

2d Lt, USAF

Committee Membership:

Dr. Douglas D. Hodson, PhD

Chair

Maj. Richard A. Dill, PhD

Member

Dr. Scott L. Nykl, PhD

Member

AFIT-ENG-MS-21-M-XXX

# Abstract

**Acknowledgments**

stuff

Chad A. Willis

**This Note is not to be included with the Acknowledgments – it is for information only: *It is prohibited to include any personal information in the following categories about U.S. citizens, DOD Employees and military personnel: social security account numbers; home addresses; dates of birth; telephone numbers other than duty officers which are appropriately made available to the general public; names, locations and any other identifying information about family members of DOD employees and military personnel.***

# Table of Contents

Page

[Abstract iv](#_Toc48512972)

[Table of Contents vi](#_Toc48512973)

[List of Figures viii](#_Toc48512974)

[List of Tables ix](#_Toc48512975)

[I. Introduction 1](#_Toc48512976)

[General Issue 1](#_Toc48512977)

[Problem Statement 1](#_Toc48512978)

[Research Objectives/Questions/Hypotheses 1](#_Toc48512979)

[Research Focus 1](#_Toc48512980)

[Investigative Questions 1](#_Toc48512981)

[Methodology 1](#_Toc48512982)

[Assumptions/Limitations 1](#_Toc48512983)

[Implications 2](#_Toc48512984)

[Preview 2](#_Toc48512985)

[II. Background 2](#_Toc48512986)

[2.1 Chapter Overview 2](#_Toc48512987)

[2.2 Rust 2](#_Toc48512988)

[2.3 Rust and Networking 3](#_Toc48512989)

[2.4 Entity Component System 3](#_Toc48512990)

[2.5 Physics of Flight Modeling 4](#_Toc48512991)

[2.6 FlightGear 4](#_Toc48512992)

[III. Methodology 4](#_Toc48512993)

[Chapter Overview 4](#_Toc48512994)

[Test Subjects 4](#_Toc48512995)

[Summary 4](#_Toc48512996)

[IV. Analysis and Results 5](#_Toc48512997)

[Chapter Overview 5](#_Toc48512998)

[Results of Simulation Scenarios 5](#_Toc48512999)

[Investigative Questions Answered 5](#_Toc48513000)

[Summary 5](#_Toc48513001)

[V. Conclusions and Recommendations 5](#_Toc48513002)

[Chapter Overview 5](#_Toc48513003)

[Conclusions of Research 6](#_Toc48513004)

[Significance of Research 6](#_Toc48513005)

[Recommendations for Action 6](#_Toc48513006)

[Recommendations for Future Research 6](#_Toc48513007)

[Summary 6](#_Toc48513008)

[Appendix 7](#_Toc48513009)

[Bibliography 8](#_Toc48513010)

[Vita 9](#_Toc48513011)

# List of Figures

Page

[Figure 1. Figure Example 3](#_Toc233011961)

# List of Tables

Page

[Table 1. Table Example 3](#_Toc233011979)

A RUST BASED, ECS PATTERNED, INTERACTIVE FLIGHT SIMULATOR

# I. Introduction

## General Issue

Text

Text

“Quoted text.”

## Problem Statement

Text

## Research Objectives/Questions/Hypotheses

Text

## Research Focus

Text

## Investigative Questions

Text

## Methodology

Text

## Assumptions/Limitations

Text

## Implications

Text

## Preview

Text

# II. Background

## 2.1 Chapter Overview

This chapter will give an overview of the necessary information to follow this thesis. This overview will provide the fundamentals of the Rust programming language. With that, basic networking, specifically with Rust, will be explained. Next, the Entity Component System (ECS) architecture, and an implementation of it, Specs, will be described. Furthermore, the physics for flight dynamics modeling will be laid out. Finally, the use of the FlightGear flight simulator will be explained.

## 2.2 Rust

Rust is a programming language that was designed for safety and speed. Rust can guarantee that a program will be memory safe, with no invalid data accesses, all while remaining just as fast as other popular programming languages. The C programming languages gives developers complete access to data inside memory, although it gives no safety precautions. The python programming language gives developers complete memory safety, but developers lose control of data inside memory. Rust combines the strengths of both C, and Python. It guarantees memory safety and the allows complete access to that memory – with exceptional performance.  
 The safety assurance of Rust is accomplished by its system of ownership and borrowing. Other languages, like Java, have a garbage collector, which manages when memory go out of scope. Rust does not have a garbage collector, it instead ownership. Ownership is

….With no safety concerns, “fearless concurrency” is given to the programmer because there is no worry of errors in parallel programming. The safety benefits of rust also come with no runtime cost.

Performance, concurrency, memory efficiency. Rust is a fast programming language – matches r even exceeds speeds of its peers. “fearless concurrency” parallel programming is free of errors that other languages have which makes concurrency be seen as difficult. “zero cost abstraction” features gained from rust have no runtime cost, safety does not mean a slower program speed. Downsides include longer compile time, compiler is strict, large language [1]. Compiler is like a safety net. The type system is smart enough to find all the urn time bugs. More time to compile, but once it does, its most likely right. Use rust for correctness. Good type system. Can be sure code is correct at compile time. Rust has no garbage collector, the compiler is the garbage collector. Ownership replaces the need for a garbage collector and assures memory safety. Borrowing references avoids data races with two pointers accessing the same memeory at the same time. This is good for performance, and less memory overhead. Community is excellent, ecosystem. Mozilla has been the driving factor behind rust. Used in firefox. Ecosystem is quickly growing, providing libraries, called crates in rust. Rust model pushes you to design your programs differently than OOP. Programs Less about execution flow and what the cpu is doing and more about how data is laid out in memory. And how data is laid out in memory . and how ownership of memory is given to different parts of the program at run time. Spend more time getting the compiler right and less time fixing bugs.

Ownership, borrowing (references), lifetimes = memory safety. There is exactly one binding to any given resource. When ownership of memory is transferred to another binding, the original binding cannot be used. Only one reference can own a piece of data at a time. References allow borrowing a piece of data for a moment [2].

Clear up these thoughts on ownership   
  
Traits…Error Handling…

## 2.3 Rust and Networking

Networking can simply be defined as two programs communicating with each other, sending and receiving data. To do this, two sockets are required at each end of communication. In this case, the type of socket being used is the Datagram socket. The protocol involving Datagram sockets is called the User Datagram Protocol (UDP). This protocol is considered “connectionless” because an open connection does not need to be maintained. Packets containing bytes of data may simply be built and sent to a destination without a connection needing to be established prior [3].

In Rust, a UdpSocket can be created and binded to an ip address and port on the local machine. After this, data can be sent and received from any other socket address. The socket created can then be connected to the other socket with its ip address and port. From here, data can be sent using the send function, and data can be received using the recv function. (code example from rust-lang)

A machines computer architecture represents bytes in a specific order. There are two byte orderings: most significant byte comes first in the ordering, and least significant byte first comes in the ordering. Most significant byte first is called Network Byte order, and least significant byte first is called Host Byte Order. When sending and receiving data, one must be aware of the order on either machine which is necessary to interpret the bytes correctly. So the order may need to be converted first before packets of data are sent from one socket to another [3].

## 2.4 Entity Component System

Networking can simply be defined as two programs communicating with each other, sending and receiving data. To do this, two sockets are required at each end of communication. In this case, the type of socket being used is the Datagram socket. The protocol involving Datagram sockets is called the User Datagram Protocol (UDP). This protocol is considered “connectionless” because an open connection does not need to be maintained. Packets containing bytes of data may simply be built and sent to a destination without a connection needing to be established prior [3].

In Rust, a UdpSocket can be created and binded to an ip address and port on the local machine. After this, data can be sent and received from any other socket address. The socket created can then be connected to the other socket with its ip address and port. From here, data can be sent using the send function, and data can be received using the recv function. (code example from rust-lang)

A machines computer architecture represents bytes in a specific order. There are two byte orderings: most significant byte comes first in the ordering, and least significant byte first comes in the ordering. Most significant byte first is called Network Byte order, and least significant byte first is called Host Byte Order. When sending and receiving data, one must be aware of the order on either machine which is necessary to interpret the bytes correctly. So the order may need to be converted first before packets of data are sent from one socket to another [3].

## 2.5 Physics of Flight Modeling

## 2.6 FlightGear

FlightGear is an open-source flight simulator application. In the beginning stage of this thesis, FlightGear was leveraged as a visual system. It was used to receive network UDP packets and display the result on the simulation.

1 Rust in action book  
2 <https://doc.rust-lang.org/1.9.0/book/>  
3 Beej’s Guide to Network Programming **-** Using Internet Sockets  
4. ECS game engine design

<http://t-machine.org/index.php/2007/09/03/entity-systems-are-the-future-of-mmog-development-part-1/>

<http://entity-systems.wikidot.com/>

<https://blog.gds-gov.tech/appreciating-rust-memory-safety-438301fee097>

Text

# III. Methodology

## Chapter Overview

The purpose of this chapter is

## Test Subjects

Text

## Summary

Text

# IV. Analysis and Results

## Chapter Overview

Text

## Results of Simulation Scenarios

Text

number Equation

Where:

X *bar* = population mean of sorties produced by squadron (LCOM)

μo = Actual number of sorties produced (Actual FY2002)

S = Standard deviation of the population mean of sorties produced (LCOM)

n = 75 replications

## Investigative Questions Answered

Text

## Summary

Text

# V. Conclusions and Recommendations

## Chapter Overview

Text

## Conclusions of Research

Text

## Significance of Research

Text

## Recommendations for Action

Text

## Recommendations for Future Research

Text

## Summary

Text

# Appendix

# Bibliography

Abalateo, Mila D. and Lee, Joni R. “Benchmarking Practices of Air Cargo Carriers: A Case Study Approach,” AFIT Thesis AFIT/GLM/LAL/93S-1, (Sep 93)

American National Standards Institute (ANSI). <http://www.ansi.org/> Accessed 15 Dec 2002.

# Vita

OPTIONAL

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **REPORT DOCUMENTATION PAGE** | | | | | | | | | *Form Approved*  *OMB No. 074-0188* |
| The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.  **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.** | | | | | | | | | |
| **1. REPORT DATE** *(DD-MM-YYYY)*  22-03-2012 | | | **2. REPORT TYPE**  Master’s Thesis | | | | | | **3. DATES COVERED** *(From – To)*  March 2010 – March 2012 |
| TITLE AND SUBTITLE  Title of Thesis in Title Case | | | | | | | | **5a. CONTRACT NUMBER** | |
| **5b. GRANT NUMBER** | |
| **5c. PROGRAM ELEMENT NUMBER** | |
| 1. **AUTHOR(S)**   Public, Joseph Q., Captain, USAF | | | | | | | | **5d. PROJECT NUMBER** | |
| **5e. TASK NUMBER** | |
| **5f. WORK UNIT NUMBER** | |
| **7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)**  Air Force Institute of Technology  Graduate School of Engineering and Management (AFIT/ENY)  2950 Hobson Way, Building 640  WPAFB OH 45433-8865 | | | | | | | | | **8. PERFORMING ORGANIZATION**  **REPORT NUMBER**  AFIT-ENY-MS-15-M-000 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**  AGENCY (spelled out)  ADDRESS  PHONE and EMAIL  ATTN: POC  (no sponsor enter: Intentionally left blank) | | | | | | | | | **10. SPONSOR/MONITOR’S ACRONYM(S)**  **AFRL/RHIQ (example)** |
| **11. SPONSOR/MONITOR’S REPORT NUMBER(S)** |
| **12. DISTRIBUTION/AVAILABILITY STATEMENT**  **DISTRUBTION STATEMENT A.** APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. | | | | | | | | | |
| **13. SUPPLEMENTARY NOTES**   This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States. | | | | | | | | | |
| **14. ABSTRACT**  (The abstract can be 200 words maximum.) | | | | | | | | | |
| **15. SUBJECT TERMS**  (Fill in with pertinent terminology related to the topic of your thesis.) | | | | | | | | | |
| **16. SECURITY CLASSIFICATION OF:** | | | | **17. LIMITATION OF**  **ABSTRACT**  **UU** | **18. NUMBER**  **OF PAGES**  19 | **19a. NAME OF RESPONSIBLE PERSON**  Name of Advisor, AFIT/ENY | | | |
| **a. REPORT**  **U** | **b. ABSTRACT**  **U** | **c. THIS PAGE**  **U** | | **19b. TELEPHONE NUMBER** *(Include area code)*  (937) 255-6565, ext xxxx (NOT DSN)  (emailname@afit.edu) | | | |
|  | | | |  |  | | **Standard Form 298 (Rev. 8-98)**  Prescribed by ANSI Std. Z39-18 | | |